

FIGURE 1 (PRIOR ART)

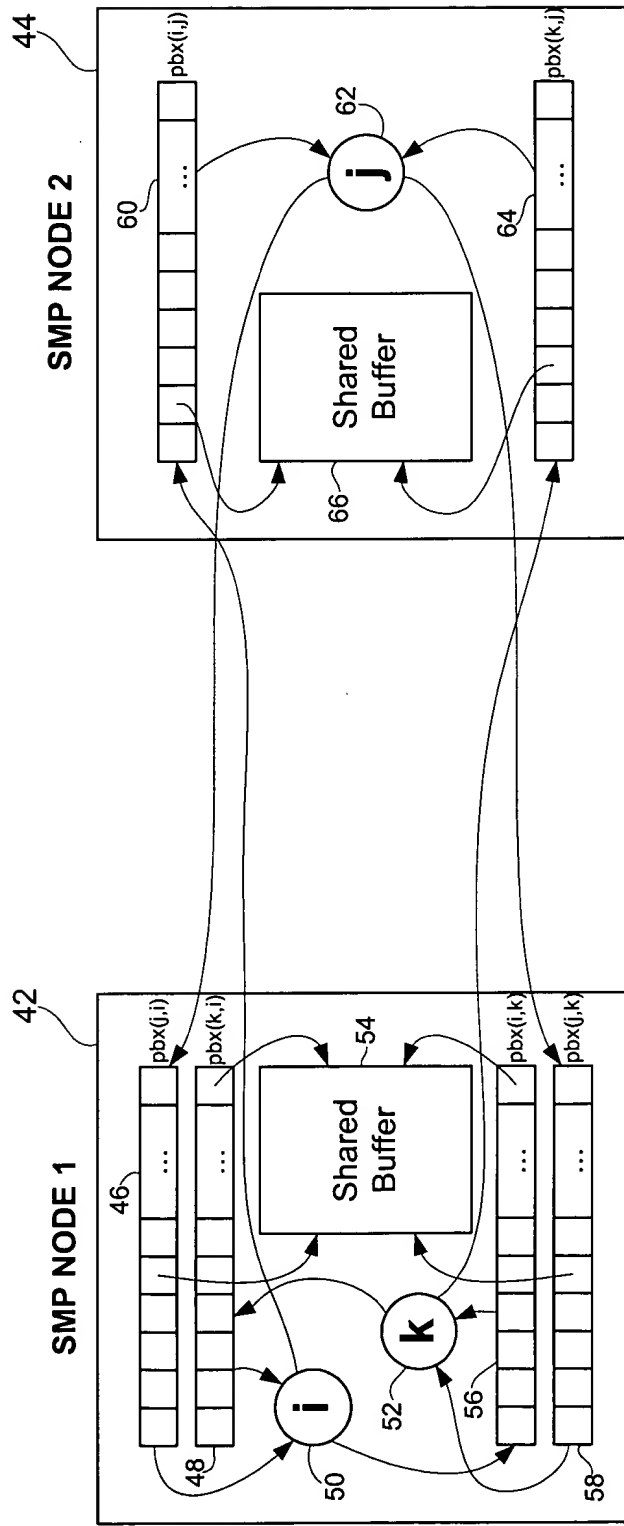


FIGURE 2 (PRIOR ART)

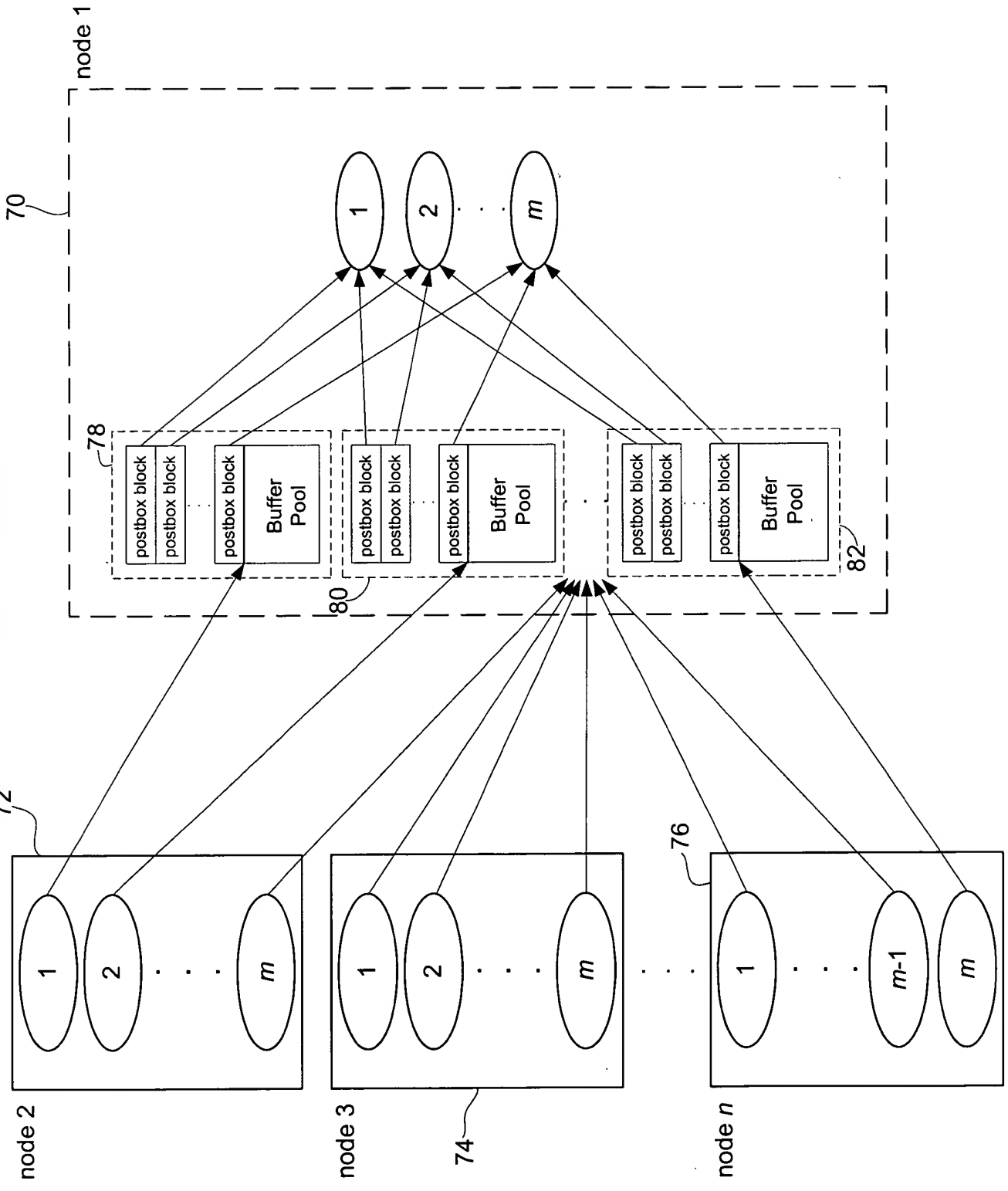


FIGURE 3 (PRIOR ART)

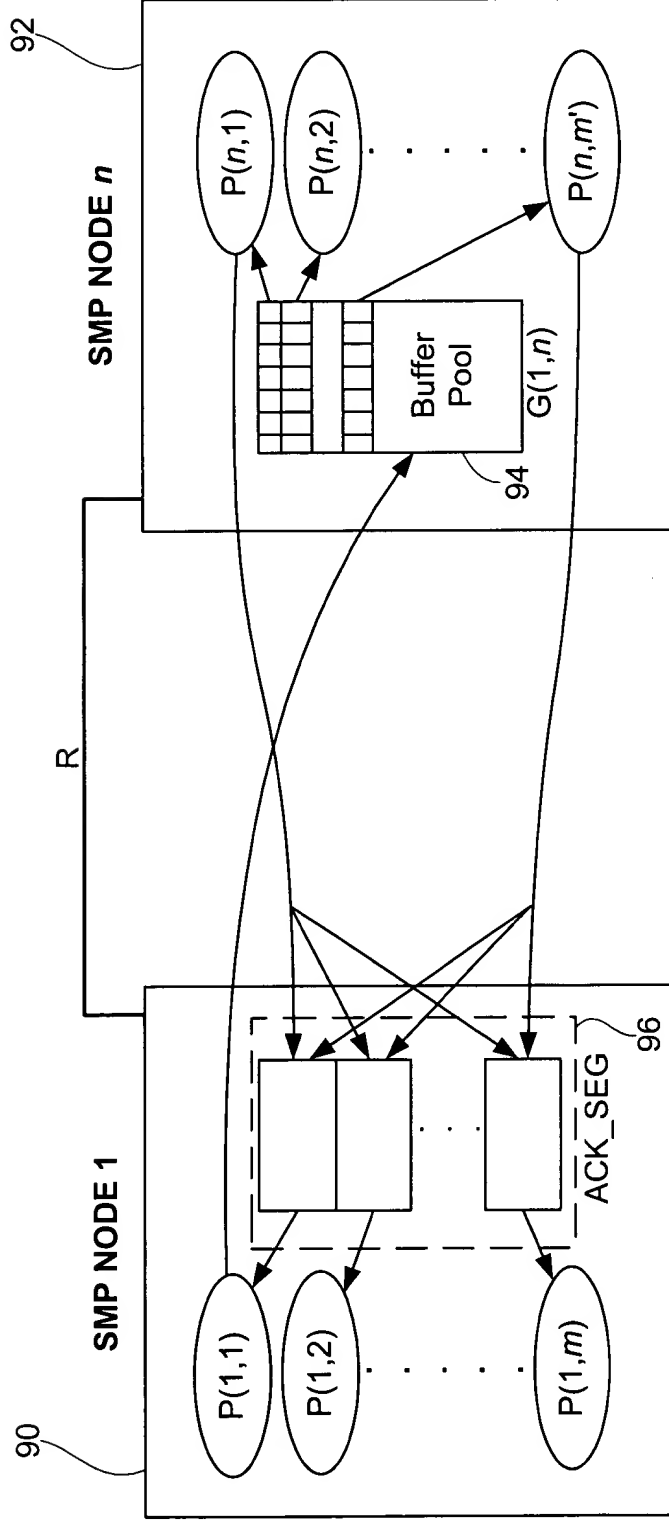


FIGURE 4

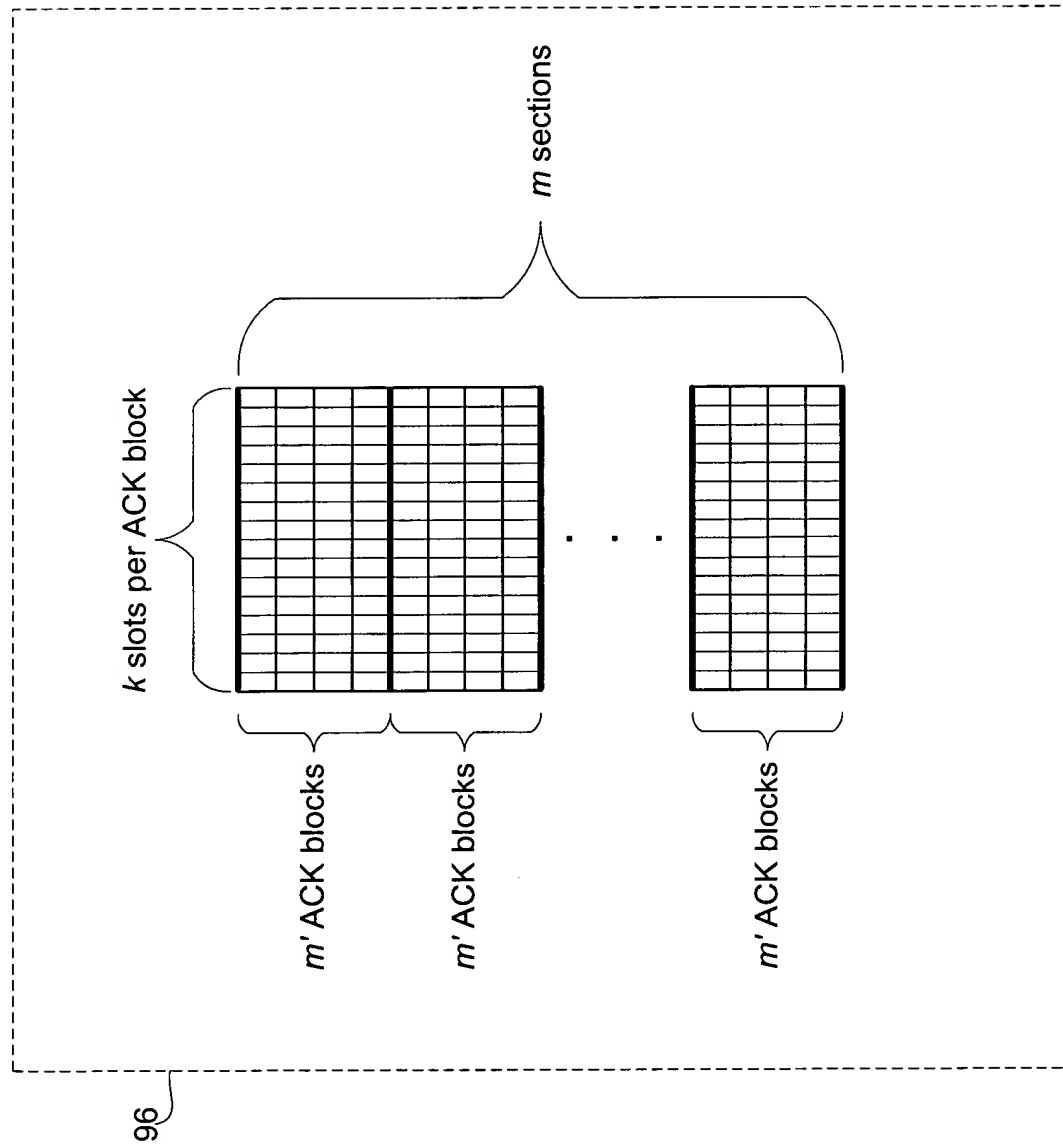


FIGURE 5

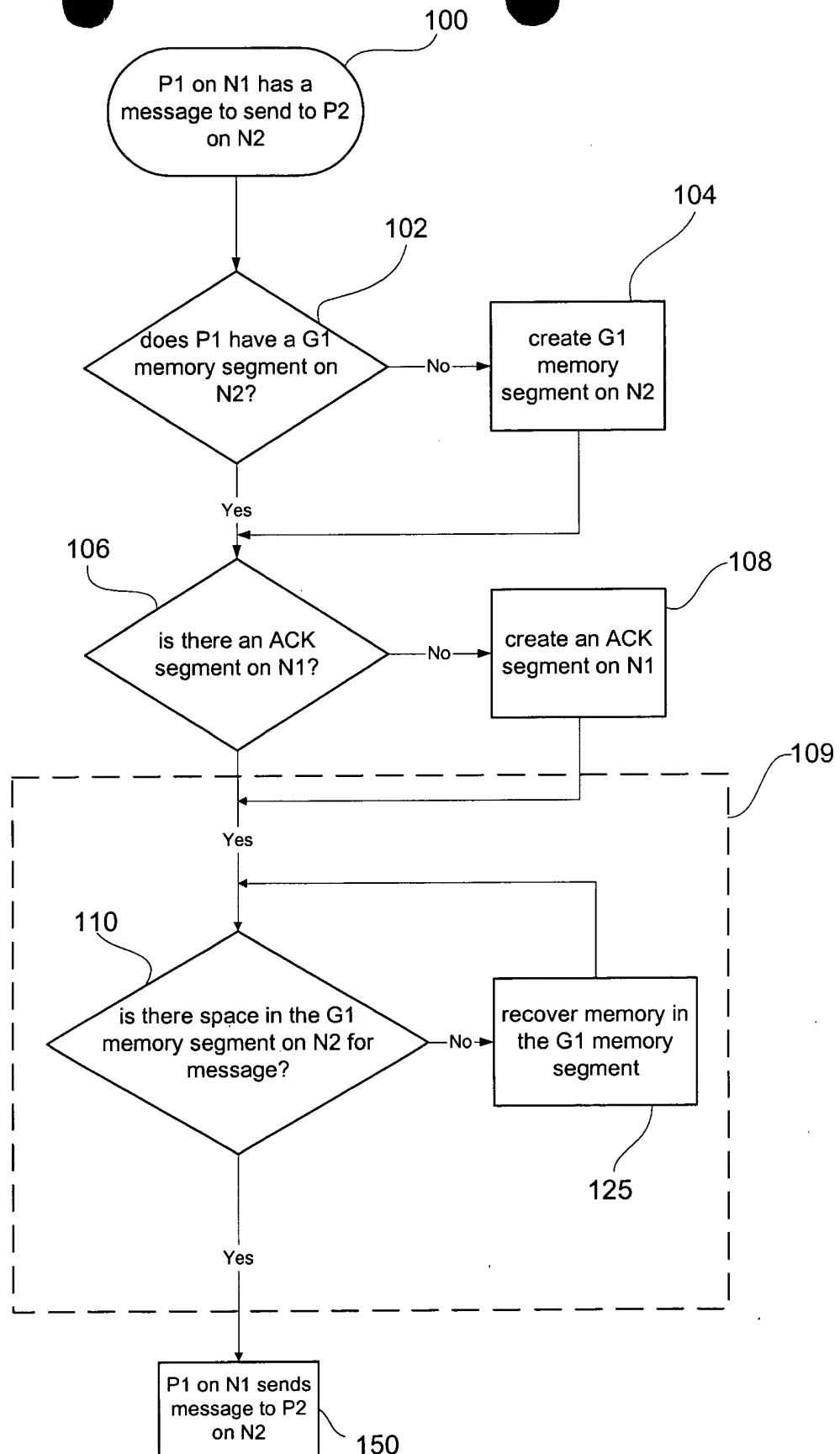
[illegible]

FIGURE 6

```

graph TD
    Entry(( )) --> 112{will current msg fit in a pbx?}
    112 -- Yes --> 114{is pbx available?}
    112 -- No --> 120{is rem. msg. size > cyclic trans. size?}
    114 -- No --> 116[reclaim pbx and buffers]
    114 -- Yes --> 126[store msg into buffers]
    120 -- Yes --> 122[transfer size = cyclic transaction size (break into smaller pieces)]
    120 -- No --> 124{are buffer blocks available?}
    122 --> 124
    124 -- Yes --> 126
    124 -- No --> 140[reclaim pbx and buffers from this receiver]
    126 --> 128[setup pointer in pbx]
    128 --> 130[write pbx]
    130 --> 132{is this rcvr. on rcvr. list?}
    132 -- Yes --> 136[decrement message size by amount sent]
    132 -- No --> 134[add receiver to receiver list]
    134 --> 136
    136 --> 138[decrement message size by amount sent]
    138 --> 139{rem. msg > 0?}
    139 -- Yes --> 124
    139 -- No --> Exit(( ))
    140 --> 142{buffer blocks available?}
    142 -- Yes --> 126
    142 -- No --> 144{anymore processes on rcvr. list?}
    144 -- Yes --> 145[pick receiver from list]
    144 -- No --> 146[reclaim pbx and buffers from receiver]
    145 --> 146
    146 --> 147{buffer blocks available?}
    147 -- Yes --> 124
    147 -- No --> 148{anymore receivers on rcvr. list?}
    148 -- Yes --> 149[next receiver]
    148 -- No --> 146
    149 --> 146
    
```

The flowchart illustrates a message transfer system with multiple receivers. It begins with a decision (112) on whether the current message fits in a pbx. If not, it checks (120) if the remaining message size is greater than the cyclic transaction size. If yes, it transfers the size (122) and checks (124) for available buffer blocks. If no, it checks (114) if the pbx is available. If no, it reclaims the pbx and buffers (116). If yes, it stores the message into buffers (126), sets up a pointer in the pbx (128), and writes the pbx (130). It then checks (132) if the receiver is on the receiver list. If yes, it decrements the message size (136). If no, it adds the receiver to the list (134). It then decrements the message size (138) and checks (139) if the remaining message is greater than 0. If yes, it checks (124) for available buffer blocks. If no, it reclaims the pbx and buffers from this receiver (140) and checks (142) for available buffer blocks. If yes, it stores the message into buffers (126). If no, it checks (144) if there are anymore processes on the receiver list. If yes, it picks a receiver from the list (145) and reclaims the pbx and buffers from the receiver (146). If no, it checks (147) for available buffer blocks. If yes, it stores the message into buffers (126). If no, it checks (148) if there are anymore receivers on the receiver list. If yes, it goes to the next receiver (149). If no, it reclaims the pbx and buffers from the receiver (146). The process ends when the remaining message is 0 (139) or when no more receivers are on the list (148).

FIGURE 7

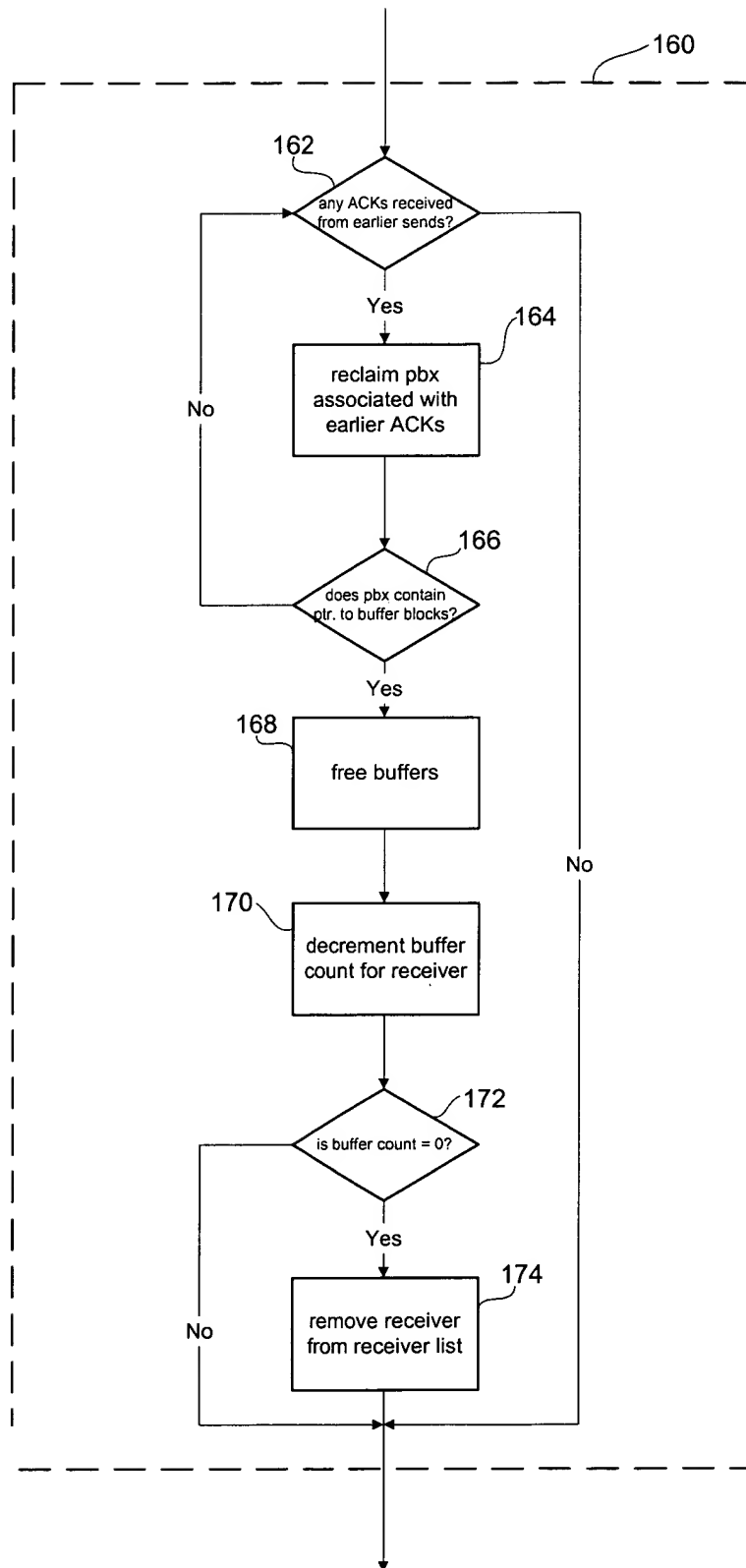


FIGURE 8